Assignment 1.2

Web Services and Cloud-Based Systems

Brunink, Yannick
`yannickbrunink@gmail.com`

Mossinkoff, Rico
`ricokoff@hotmail.com`

April 7, 2020

RESTFUL service

Although SOAP is still widely used, a lot of web-companies have switched or are switching to another web-service design model: REST. Therefore, the structure of this assignment follows the natural timeline by also first creating a SOAP service and then switching to REST. In this part of the assignment the service is a REST "URL-shortener".

The URLshortener service

For the URLshortener service we used the REST packages from the Java API for XML Web Services (JAX-WS). These include easy functionality to create functions for specific HTTP requests. First we created a maven web project to get some of the necessary files. For the shortURL we used a standard format ("lin.k******") where we generated a unique random alphanumeric value with a length of 6. We created functions for all the HTTP requests, and implemented them accordingly. For the GET request and DELETE request we created 2 functions. One version expects parameters while the other does not. To save shortURLID's, we used 2 hashmaps. We made mappings from the URL to the shortURLID's and the other way around. This makes it easier to look up both sides of the mapping, which we do with different HTTP requests.

When sending a POST request we first check if the URL is valid using a regular expression. Then we check if the URL is already known. If it is known we return the corresponding shortURL. Otherwise we generate a new one, save it in the HashMaps and return it. With the GET requests we either return the URL of a shortURLID or we return all shortURLIDs. PUT replaces the URL at a given shortURLID and DELETE either deletes a given shortURLID mapping or it deletes all the mappings.

All the functions have some error handling implemented as well. Normally the service should send an error code to the service when something went wrong. We decided to this as a string, so we could print the status code at the client side. This made sure that we did not get a default error page (e.g. 404 NOT FOUND) and that we could also display the error code when the request was valid. This could be easily changed by returning a Response type with a body message instead of a String.

The URLshortener Client

After the web service is started using TomCat, the client can just be ran as a normal java file. It waits for the users input and performs the given commands. It has functions for every request it can make. This way the user can easily test the web service interactively, which speeds up the process.

Question 3. How would you implement URL-shortener for multiple users?

If there are multiple users that each want their own shortURLID mappings, we can make use of a userID. We can use a hashmap to map userIDs to the hashmaps they use. The user includes his userID with every request he makes, so the server can load the corresponding HashMaps. REST services should be stateless, so we do no save a user session. As the client provides the userID with every request, we keep the stateless property while still offering multi-user functionality.